Department of Computer Science and Engineering
Institute of Technical Education & Research, SOA, Deemed to be University

MAJOR ASSIGNMENT-2
UNIX Systems Programming (CSE 3041)

Part 1: Below program 1 creates a chain of processes using fork system call like below diagram. It takes a single command-line argument that specifies the number of processes to create. Before exiting, each process outputs its i value, its process ID, its parent process ID and the process ID of its child. The parent does not execute wait. If the parent exits before the child, the child becomes an orphan. In this case, the child process is adopted by a special system process (which traditionally is a process, init, with process ID of 1). As a result, some of the processes may indicate a parent process ID of 1.

Program 1: Chain of Process

```
int main (int argc, char *argv[]) {
pid_t childpid = 0;
int i, n;
if (argc != 2){
fprintf(stderr, "Usage: %s processes\n", argv[0]);
return 1; }
n = atoi(argv[1]);
for (i = 1; i < n; i++)
if (childpid = fork())
break;
fprintf(stderr, "i:%d process ID:%ld parent ID:%ld child ID:%ld\n", i, (long)getpid(),
(long)getppid(), (long)childpid);
return 0;
}
```

Fig. 1

Q1. Run above program and observe the results for different numbers of processes.

Q2. Fill in the actual process IDs of the processes in the Fig. 1 for a run with command-line argument value of 4.

Q3. Experiment with different values for the command-line argument to find out the largest number of processes that the program can generate. Observe the fraction that are adopted by init.

Q4. Place sleep(10); directly before the final fprintf statement in Program 1. What is the maximum number of processes generated in this case?

Q5. Put a loop around the final fprintf in Program 1. Have the loop execute k times. Put

sleep(m) inside this loop after the fprintf. Pass k and m on the command line. Run the program for several values of n, k and m. Observe the results.

Q6. Modify Program 1 by putting a wait function call before the final fprintf statement. How does this affect the output of the program?

Q7. Modify Program 1 by replacing the final fprintf statement with four fprintf statements, one each for the four integers displayed. Only the last one should output a newline. What happens when you run this program? Can you tell which process generated each part of the output? Run the program several times and see if there is a difference in the output.

Part 2:

Q1. Develop a C code to create a fan of n = 4 processes by calling fork() in a loop as per the below structures. Display the process ID and parent ID in the order 1, 2, 3, and 4. Run your code for different values of n and check the exact order.